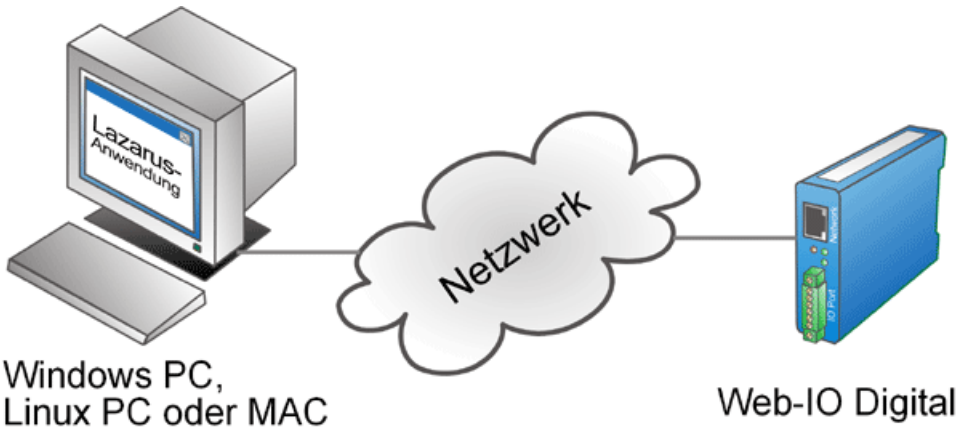
Applikation zum Web-IO Digital:

Web-IO Digital mit Lazarus/Free Pascal steuern und überwachen

Produktübersicht
Applikationsübersicht

Lazarus ist eine OpenSource Entwicklungsumgbung, die Delphi emuliert und auf Free Pascal aufsetzt. Das interessante an Lazaur ist, das diese Entwicklungsumgebung für Windows, Linux und einige MAC Systeme verfügbar ist und die Projekte übergreifend nutzbar sind. Unter https://www.lazarus.freepascal.org/ kann Lazarus kostenlos herunter geladen werden.

Als einfach zu erlernende Hochsprache bietet Lazarus fast alles, was zum Programmieren von Web-IO® Anwendungen nötig ist. Für die Abwicklung der TCP/IP-Kommunikation muss allerdings das zusätzliche LNET Steuerelemt als Komponentenpaket installiert werden.



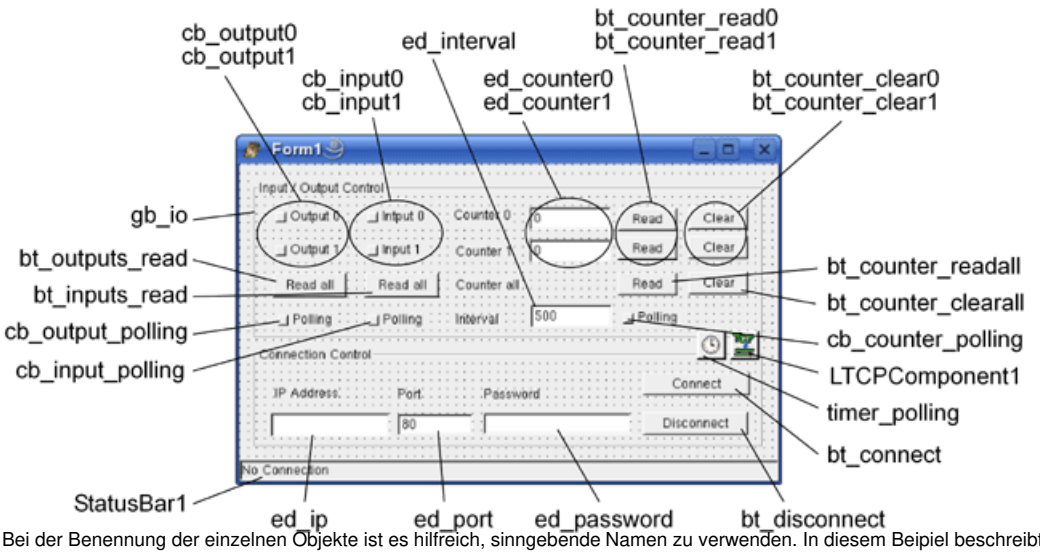
Mit dem folgenden Programmbeispiel können Sie Ihr Web-IO Digital mit seinen Inputs und Outputs in einer Windows-Anwendung abbilden. Darüber hinaus können Sie die Outputs des Web-IO schalten.

Vorbereitungen

Sie haben Ihr Web-IO Digital bereits

- mit Strom versorgt
- Eingänge und Ausgänge beschaltet
- an Ihr Netzwerk angeschlossen
- mit einer IP-Adresse versehen - mit WuTility kein Problem

1. Zusammenstellen der verschiedenen Bedienelemente und Anzeigeobjekte im Lazarus-Form



Bei der Benennung der einzelnen Objekte ist es hilfreich, sinngebende Namen zu verwenden. In diesem Beipiel beschreibt der erste Teil des Namens die Art des Objektes und der zweite Teil die Funktion.

2. Die Verbindungskontrolle

Einleiten der Verbindung
Durch Eingabe der IP-Adresse des Web-IO in das Textfeld ed_ip und Klick auf den Button bt_connect wird der Verbindungsaufbau gestartet.

```
procedure TForm1.bt_connectClick(Sender: TObject);
begin
 if ed_ip.Text <> '' then
 begin
  bt_connect.Enabled := false;
  LTCPComponent1.Connect(ed_ip.Text, strtoint(ed_port.Text));
 end;
end;
```

Verbindung kommt zustande
Sobald das Web-IO die Verbindung annimmt, führt das ClintSocket-Steuerelement die entsprechende Prozedur aus. In der Statuszeile wird das Zustandekommen der Verbindung angezeigt, die Bedienelemente werden zur Benutzung freigegeben und der Disconnect-Button wird bedienbar.

```
procedure TForm1.LTCPComponent1Connect(aSocket:TLSocket);
begin
 StatusBar1.SimpleText :='Connected to ' + ed_ip.Text;
 bt_connect.Enabled := False;
 bt_disconnect.Enabled := True;
 gb_io.Enabled := True;
end;
```

Trennen der Verbindung
Die Verbindung bleibt solange bestehen, bis sie vom Benutzer durch Klick auf den Disconnect-Button beendet wird oder das Web-IO die Verbindung beendet.

```
procedure TForm1.bt_disconnectClick(Sender:TObject);
begin
 StatusBar1.SimpleText :='No Connection';
 bt_connect.Enabled := True;
 bt_disconnect.Enabled := False;
 gb_io.Enabled := False;
 LTCPComponent1.Disconnect;
end;
```

Auch in diesem Fall ruft das ClientSocket-Steuerelement eine entsprechende Prozedur auf

```
procedure TForm1.LTCPComponent1Disconnect(aSocket:TLSocket);
begin
 StatusBar1.SimpleText :='No Connection';
 bt_connect.Enabled := True;
 bt_disconnect.Enabled := False;
 gb_io.Enabled := False;
 LTCPComponent1.Disconnect;
end;
```

Verbindungsfehler
Auch im Fall eines Verbindungsfehlers führt das ClientSocket-Steuerelement eine entsprechende Prozedure aus, die im Groben der Disconnect Prozedur entspricht. In der Statuszeile wird zusätzlich die Winsock-Fehlernummer mit ausgegeben und der ErrorCode wird abschließend auf 0 gesetzt, damit es nicht zu einem Laufzeitfehler kommt.

```
procedure TForm1.LTCPComponent1Error(const msg: string; aSocket: TLSocket);
begin
 StatusBar1.SimpleText :='Error: '+ msg;
 bt_connect.Enabled := True;
 bt_disconnect.Enabled := False;
 gb_io.Enabled := False;
 LTCPComponent1.Disconnect;
end;
```

3. Bedienung und Kommunikation von Client-Seite

Sobald eine Verbindung mit dem Web-IO zustande gekommen ist, kann der Anwender durch Bedienung der entsprechenden Programmelemente Kommandos an das Web-IO senden

Setzen der Outputs
Das Setzen der Outputs wird dem Anwender über zwei Checkboxen cb_outputx ermöglicht. Das Programm nutzt dazu das MouseUP-Ereignis dieses Objektes. Wird ein Mouse Up, also ein Loslassen der Output-Checkbox registriert, führt das Programm die entsprechende Prozedur aus und gibt - je nach dem ob die Checkbox gesetzt ist oder nicht - das passende Kommando an das Web-IO weiter.

```
procedure TForm1.cb_output0MouseUp(Sender:TOBject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 if cb_output0.Checked then
 LTCPComponent1.SendMessage('GET/outputaccess0?PW=' + ed_password.Text+ '&State=ON&',nil)
 else
 LTCPComponent1.SendMessage('GET/outputaccess0?PW=' + ed_password.Text + '&State=OFF&',nil)
end;


procedure TForm1.cb_output1MouseUp(Sender:TOBject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 if cb_output1.Checked then
 LTCPComponent1.SendMessage('GET/outputaccess1?PW=' + ed_password.Text + '&State=ON&',nil)
 else
 LTCPComponent1.SendMessage('GET/outputaccess1?PW=' + ed_password.Text + '&State=OFF&',nil)
end;
```

Output/Input-Status abfragen
Den Status der Outputs und Inputs kann der Anwender durch anklicken des zugehörigen Buttons anfordern.

```
procedure TForm1.bt_outputs_readClick(Sender:TObject);
begin
 LTCPComponent1.SendMessage('GET/output?PW=' + ed_password.Text + '&',nil);
end;


procedure TForm1.bt_inputs_readClick(Sender:TObject);
begin
 LTCPComponent1.SendMessage('GET/input?PW=' + ed_password.Text + '&',nil);
end;
```

Counter abfragen löschen
Auch die Zählerstände der Input-Counter lassen sich abfragen bzw. löschen.

```
procedure TForm1.bt_counter_read0Click(Sender:TObject);
begin
 LTCPComponent1.SendMessage('GET/counter0?PW=' + ed_password.Text + '&',nil);
end;


procedure TForm1.bt_counter_read1Click(Sender:TObject);
begin
 LTCPComponent1.SendMessage('GET/counter1?PW=' + ed_password.Text + '&',nil);
end;


procedure TForm1.bt_counter_clear0Click(Sender: TObject);
begin
 LTCPComponent1.SendMessage('GET/counterclear0?PW=' + ed_password.Text + '&',nil);
end;


 procedure TForm1.bt_counter_clear1Click(Sender:TObject);
 begin
  LTCPComponent1.SendMessage('GET/counterclear1?PW=' + ed_password.Text + '&',nil);
 end;
```

Natürlich lassen sich auch alle Counter zeitgleich lesen bzw. löschen.

```
procedure TForm1.bt_counter_readallClick(Sender:TObject);
begin
 LTCPComponent1.SendMessage('GET/counter?PW=' + ed_password.Text + '&',nil);
end;


procedure TForm1.bt_counter_clearallClick(Sender:TObject);
begin
 LTCPComponent1.SendMessage('GET/counterclear?PW=' + ed_password.Text + '&',nil);
end;
```

4. Datenempfang vom Web-IO

Auswerten und Anzeigen der empfangenen Daten

- Alle Kommandos und Anfragen an das Web-IO werden mit einem Antwort-String quittiert. Dabei haben die Antworten je nach Type einen spezifischen Aufbau.
- Für die Outputs: output;<Binärwert des Outputstatus im hexadezimalen Format>
- Für die Inputs: input;<Binärwert des Outputstatus im hexadezimalen Format>
- Für die Counter: counterx;<dezimaler Zählerstand>
- oder counter;<dezimaler Zählerstand 0 >; <dezimaler Zählerstand 0 >; ..... wenn alle Counter auf einmal gelesen werden sollen.
- Alle Antwort-Strings sind mit einem 0-Byte abgeschlossen.
- Werden vom ClientSocket Steuerelement Daten empfangen, ruft dieses die entsprechende Prozedur auf

```
procedure TForm1.LTCPComponent1Receive(aSocket:TLSocket);
var ReceiveString : String;
SemiPosition : integer;
OutputValue : word;
InputValue : word;
begin
 aSocket.GetMessage(ReceiveString);
 if Receivestring[1] = 'o' then
 begin
  OutputValue := HextoInt(copy(ReceiveString, pos(';',ReceiveString)+1,length(ReceiveString)-pos(';',ReceiveString)-1));
 if OutputValue and 1 = 1 then
  cb_output0.Checked := True
 else
  cb_output0.Checked := False;
 if OutputValue and 2 = 2 then
  cb_output1.Checked := True
 else
  cb_output1.Checked := False;
 end;
 if Receivestring[1] = 'i' then
 begin
  InputValue := HextoInt(copy(ReceiveString, pos(';',ReceiveString)+1, length(ReceiveString)-pos(';',ReceiveString)-1));
 if InputValue and 1 = 1 then
  cb_input0.Checked := True
 else
  cb_input0.Checked := False;
 if InputValue and 2 = 2 then
  cb_input1.Checked := True
 else
  cb_input1.Checked := False;
 end;
 if Receivestring[1] = 'c' then
 begin
 if copy(ReceiveString, 8, 1) = '0' then
  ed_counter0.Text := copy(ReceiveString, 10,length(ReceiveString)-10);
 if copy(ReceiveString, 8, 1) = '1' then
  ed_counter1.Text := copy(ReceiveString, 10,length(ReceiveString)-10);
 if copy(ReceiveString, 8, 1) = ';' then
 begin
  ReceiveString[8] := ' ';
  ed_counter0.Text := copy(ReceiveString, 9, pos(';',ReceiveString)-9);
  ed_counter1.Text := copy(ReceiveString, pos(';',ReceiveString)+1, length(ReceiveString)-pos(';',ReceiveString)-1);
 end;
 end;
end;
```

Die Empfangsprozedur überprüft anhand des ersten Zeichens der Empfangsdaten, ob es um Input, Output oder Counter Meldungen geht. Abhängig davon wird z.B. festgestellt, welcher Output welchen Status hat. Da der Output-Status hexadezimal übergeben wird, ist zunächst eine Wandlung in einen Integerwert nötig, um weitere Schritte zu berechnen. Zur Wandlung wurde eine entsprechende Funktion in das Programm eingebunden. Bei den Countern ist es sowohl möglich, einzelne Zählerwerte abzufragen, als auch alle Counter in einem Zug auszulesen. Die einzelnen Zählerstände werden dann dezimal mit Semikolon getrennt in einem String ausgegeben

```
function HexToInt(HexString :String) : longWord;
var CharCount : integer;
 HexCharValue : longWord;
 HexValue : longWord;
begin
 HexValue := 0;
 HexString := UpperCase(HexString);
 for CharCount := 1 to length(HexString)
do
 begin
 HexCharValue := ord(HexString[CharCount]);
 if HexCharValue > 57 then
  HexCharValue := HexCharValue - 55
 else
  HexCharValue := HexCharValue - 48;
  HexCharValue := HexCharValue
shl ((length(HexString) - CharCount) * 4);
  HexValue := HexValue or HexCharValue;
 end;
 HexToInt := HexValue;
end;
```

Zyklisches Abfragen bestimmter Werte
Um auch eine automatische Aktualisierung der Anzeige zu ermöglichen, wird ein Timer benutzt.

In Abhängigkeit der Checkboxen für Output-, Input- und Counter-Polling werden die entsprechenden Informationen im eingestellten Intervall vom Web-IO abgerufen.

5. Polling

Das gewünschte Intervall kann in das entsprechende Textfeld eingegeben werden.

```
procedure TForm1.timer_pollingTimer(Sender:TObject);
begin
 if LTCPComponent1.Connected and cb_output_polling.Checked then
  LTCPComponent1.SendMessage('GET/output?PW=' + ed_password.Text + '&',nil);
 if LTCPComponent1.Connected and cb_input_polling.Checked then
  LTCPComponent1.SendMessage('GET/input?PW=' + ed_password.Text + '&',nil);
 if LTCPComponent1.Connected and cb_counter_polling.Checked then
  LTCPComponent1.SendMessage('GET/counter?PW=' + ed_password.Text + '&',nil);
end;
```

Bei Änderung wird das Timer-Intervall dann automatisch angepasst.

```
procedure TForm1.ed_intervalChange(Sender:TObject);
begin
 timer_polling.Interval:= strtoint(ed_interval.text);
end;
```

Das Beispiel Programm unterstützt alle gängigen Funktionen des Web-IO im Kommando-String Modus, optimiert für das Web-IO 2x Digital Input, 2x Digital Output. Für die anderen Web-IO Modelle müssen ggf. Anpassung am Programm vorgenommen werden. Weitere Programmbeispiele zur Socket-Programmierung finden Sie auf den Tool-Seiten zum Web-IO. Eine Detaillierte Beschreibung zur Socketschnittstelle der Web-IO Digital Modelle finden Sie im Referenzhandbuch.

Programmbeispiel herunterladen

W&T
www.WuT.de

Wir sind gerne persönlich für Sie da:

Wiesemann & Theis
GmbH
Porschestr. 12
42279 Wuppertal
Tel.: 0202/2680-110 (Mo-Fr. 8-17 Uhr)
Fax: 0202/2680-265
info@wut.de